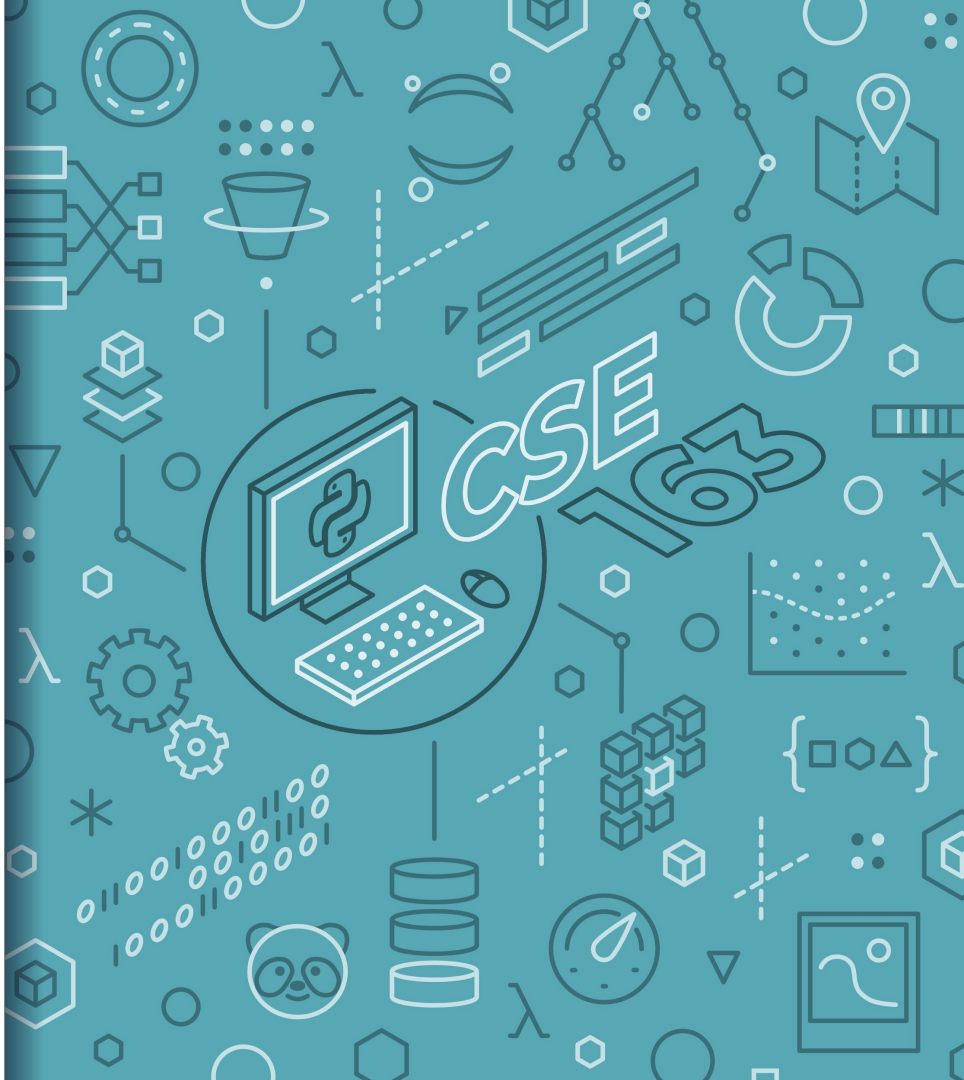
CSE 163

Syllabus

Introduction to Python

Hunter Schafer



Agenda

- Overview
  - What is this class? Who is taking this class?
- Who are we?
- Syllabus
- Development Environment
- Python Crash Course - Day 1
  - Hello world
  - REPL vs. Script
  - Variables and Arithmetic

Overview

What is this class?

Competencies

1. **More advanced programming concepts** than in CSE 142 or CSE 160 including how to write bigger programs with multiple classes and modules.

2. How to **work with different types of data**: tabular, text, images, geo-spatial, etc.

3. Ecosystem of **data science tools** including Jupyter Notebook and various **data science libraries** including scikit-image, scikit-learn, and pandas data frames.

4. Basic concepts related to code complexity, efficiency of different types of data structures, and memory management.

Overview

Who is taking this class?

This class is designed to have students from

- **142**: Know control structures, file I/O, arrays in Java
  - Will spend first weeks learning 142 in Python fast!
  - Practice is **KEY**!
- **160**: Know control structures, file I/O, data structures in Python
  - First week will be review while everyone learns Python
- **143 or Beyond**: Seen more advanced programming in Java
  - Class material should be **complementary** to what you would have learned in 143
  - **Competency 1** is aimed at the 143 level of programming

Who am I?

- Hunter Schafer, Lecturer
- Office Hours
  - Time: 10:30 am - 12:30 pm, Wednesdays
  - Location: CSE 444
- Contact
  - Personal Matters: hschafer@cs.washington.edu
  - Course Content + Logistics: Piazza

Who are the TAs?

AA (12:30)

Erika Wolfe
eywolfe@cs

Josh Ervin
joshue@cs

AB (1:30)

Dylan Jergens
dylanj7@cs

Erik Hoberg
ehoberg@cs

AC (2:30)

Nicole Riley
nriley16@cs

Joely Nelson
joelyn@cs

Syllabus

Mon	Tue	Wed	Thur	Fri
Lecture	Nothing 😢	Lecture	Section	Lecture
			Prev HW **Due**	Next HW **Released** Project Checkpoints

- We don't record attendance in lecture/section, but attending these sessions is expected
- Panopto for Lecture (on Canvas)

Homeworks + Project

With the scaffolding from **1 and 2**, you are probably now capable to tackle the homeworks. These will be complex and challenging, but you'll continue to **learn by doing.**

Sections

Practice material covered in **1** in a context where a TA can help you.

The emphasis is still on you l**earning by doing.**

Lectures

Introduced to material for the first time. Mixed with activities and demos to give you a chance to **learn by doing.**

No where near mastery yet!

Assessment

Your learning from this course will be assessed by:

- **Weekly Homework Assignments**
  - **Weight**: 60%
  - **Number**: Approximately 7
- **Exams**
  - **Weight**: 25%
  - **Number**: 2 Exams
- **Final Project**
  - **Weight**: 15%
  - **Number**: Just one project, but multiple check-ins

Homework Logistics

- **Late Days**
  - 5 Free Late Days for the whole quarter.
  - Can use up to 2 Late Days on any assignment.
  - Each Late Day used after the 5 Free Late Days results in a -5% on that assignment

- **Collaboration**
  - You are encouraged to discuss assignments and concepts **at a high level**
    - If you have code in front of you in your discussion, probably **NOT** high level
  - All code and answers submitted must be your own
  - Project can be done in groups of 2

- **Getting Help**
  - Piazza (link with add code in Syllabus)
  - Office Hours

Project

- Culmination of all the things you learned in this class.
- Open ended project where you find and use real-world datasets to answer an interesting question.
- Broken into various checkpoints throughout quarter:
  - Find some possible ideas for datasets and questions
  - Pick a research question and your datasets + find a partner
  - Outline methodology and define work plan
  - Gather results and write final report
- Final Project presentations during final exam slot

Python Crash Course

Day 1

Hello World!

- Write a file called hello.py that prints “Hello World!”

Attempt 1

```
print('Hello World!')
```

Attempt 2

```
def main():
    print('Hello World!')

if __name__ == '__main__':
    main()
```

Python Modes

- Script
  - Write a .py file and run it
    `python my_file.py`
  - Runs file from top to bottom
- Interactive Shell
  - Read Evaluate Print Loop (REPL)
  - Interactive mode to try small chunks of code
  - Simple: `python`
  - Complex: Jupyter Notebook
- Follow Software instructions to set up your computer for development

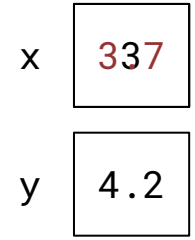
Your HW + Project

Lecture (usually)

Variables

- A **variable** is a named box that stores a value
- Variables are defined as: `<name> = <value>`
- Examples

```
x = 3
y = 4.2
x = 3.7
```



- A variable can only hold a single value* at any point in time, but the same variable can hold different types of data throughout the whole program
  - We call Python a **dynamically typed** programming language because the type of a variable can change

* We will see how to store more than one thing at a time later

Expressions

- Right hand side doesn't have to be a simple value (e.g. `4.2`), but can be an **expression** that *evaluates* to a simple value
- You probably know most operators for expressions from your programming background.

```
a = 2 + 3          # addition
b = a - 1          # subtraction
c = (b + a) * 2    # nested, multiplication
d = 3 / 2          # division
e = 3 // 2         # integer division
f = a % b          # mod
g = b ** 2         # exponentiation
```

a	5	b	4	c	18		
d	1.5	e	1	f	1	g	16

Poll Everywhere

- **Goal**: Get you actively participating in your learning
- Typical Activity
  - Question is posed
  - **Think** (1 min): Think about the question on your own
  - **Pair** (2 min): Talk with your neighbor to discuss question
    - If you arrive at different conclusions, discuss your logic and figure out why you differ!
    - If you arrived at the same conclusion, discuss why the other answers might be wrong!
  - **Share** (1 min): We discuss the conclusions as a class

- During each of the **Think** and **Pair** stages, you will respond to the question via a Poll Everywhere poll
  - The poll will only be open for the last **15** seconds of each of the stage
  - Not worth any points, just here to help you learn!

Poll Everywhere

Think

1 minute

pollev.com/cse163

Make a prediction of what this program will output.

Note that `print(1, 2, 3)` outputs: 1 2 3

```
x = 5
y = 7
z = y - x

x = 2
y = z / 2
print(x, y, z)
```

Options

- 2 2.5 5
- 2 1.0 2
- 2 2 2
- 2 1 2

Poll Everywhere

Pair

1 minute

pollev.com/cse163

Make a prediction of what this program will output.

Note that `print(1, 2, 3)` outputs: 1 2 3

```
x = 5
y = 7
z = y - x

x = 2
y = z / 2
print(x, y, z)
```

Options

- 2 2.5 5
- 2 1.0 2
- 2 2 2
- 2 1 2

Next Time

- Jupyter Notebooks
- Python Crash Course - Day 2
  - Conditionals
  - Functions, parameters, returns
  - Strings
- Documenting code
- Testing code

Before Next Time

- Fill out begin of quarter survey
  - Worth 2 points
  - Due @ 1 pm, no late days
- Setup software for course
- Try practice problems from today

bool
True/False

- A bool has two values: True, False
- Can you logical operators: and, or, not

```
b1 = False
b2 = True
print(b1 and b2)       # False
print(b1 or b2)        # True
print(not b2 or b1)    # False
print(not (b2 or b1))  # False
```

- Can get bools by comparing numbers

```
x = 3
print(x < 4)  # True
print(x >= 5) # False
print(x == 2) # False
```

Casting + Types

- Every piece of data in Python has a **type**. You can convert between types by **casting**.

```
x = 1.4
print(x)          # 1.4
print(int(x))     # 1

x = "1.7"
print(float(x))   # 1.7
print(int(x))     # Error
```

- Commonly used types: `int`, `float`, `bool`, `str`, `char`
- Can use `type(x)` to find x's type

While Loop

- A **while loop** has a **condition** and a **body**. It executes the body repeatedly until the condition is false.

```
x = 1
while x < 100:
    print(x)
    x = x * 2
print('After loop', x)
```

```
1
2
4
8
16
32
64
After loop 128
```

- **Important**: Python uses indentation to determine what belongs inside the loop!
  - Very common beginner error
    `IndentationError: unexpected indent`

For loop

- If you know Java, for loops in Python look pretty different
- We won't give a formal definition of a for loop at this time, but will start with a simple example

```
for i in range(5):
    print(i)
print('After loop', i)
```

```
0
1
2
3
4
After loop 4
```

- The many uses of `range`
  - `range(A)`
    - Numbers between 0 inc. and A exc.
  - `range(A, B)`
    - Numbers between A inc. and B exc.
  - `range(A, B, C)`
    - Numbers between A inc. and B exc. (steps by C)

Poll Everywhere

Pair

2 minutes

pollev.com/cse163

Make a prediction of what this program will output.

Note: `j -= 1` *is the same thing as* `j = j - 1`

```
for i in range(2, 11, 3):
    j = i - 1
    while i >= 1 and i % j != 0:
        j -= 1
    print(i, ':', j)
```

Options (new line separated by /)

- 2:2/5:5/8:8
- 2:2/5:5/8:8/11:11
- 2:1/5:1/8:4
- 2:1/5:1/8:4/11:1
- 2:1/5:4/8:7/11:10